Work1—6:

1. Какое представление позволяет просмотреть файловую структуру проекта,

включая скрытые файлы?

В Android Studio для этого используется представление "Project", при этом для отображения скрытых файлов нужно выбрать соответствующую настройку отображения (например, выбрав "Show Hidden Files" в настройках).

2. Какая структура папок (групп) в представлении Android View?

В представлении "Android" папки сгруппированы для удобства работы с проектом. Основные группы:

manifests — содержит AndroidManifest.xml.

java — исходный код Java или Kotlin.

res — ресурсы проекта (layout, drawable, values и т. д.).

assets — дополнительные файлы, которые могут быть использованы в приложении.

build — скомпилированные файлы проекта.

3. Какая структура папок в представлении Project View?

В представлении "Project" файловая структура представлена в виде обычных папок и файлов, и может включать такие папки, как:

app — основная папка с исходным кодом и ресурсами.

gradle — файлы конфигурации сборки.

libs — библиотеки.

build — результат сборки. При этом представление может быть настроено на отображение по разным фильтрам (например, "Project", "Packages" и другие).

4. Может ли пакет apk содержать несколько файлов AndroidManifest.xml? Почему?

Нет, пакет APK не может содержать несколько файлов AndroidManifest.xml. В APK всегда используется только один файл манифеста, который описывает структуру приложения, его компоненты и требования. Этот файл манифеста объединяет все необходимые данные для работы приложения.

5. Может ли проект Android Studio несколько файлов манифестов? Каких?

Да, проект Android Studio может содержать несколько файлов манифестов. Это особенно актуально, если проект поддерживает несколько вариантов сборки (например, для разных типов устройств или вариантов языков). Например, можно использовать разные файлы манифестов для:

Основного манифеста (в директории src/main/).

Специфичных для определенных флагов сборки манифестов, например, для версий с разными возможностями или настройками для разных продуктов (src/debug/, src/release/ и т. д.).

6. Какой элемент в файле manifest.xml является корневым? Какой обязательный

элемент он должен содержать?

Корневым элементом в файле AndroidManifest.xml является <manifest>.

Этот элемент должен содержать обязательный атрибут package, который указывает на уникальный идентификатор пакета приложения, например:

xml

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.myapp">

7. Что определяет атрибут xmlns:android? Является ли он обязательным?

Атрибут xmlns:android задает пространство имен для всех атрибутов Android в XML-файле. Он указывает, что все атрибуты, начинающиеся с android:, принадлежат Android-системе и используются для задания настроек и параметров компонентов приложения.Да, этот атрибут обязательный для файлов манифеста и других XML-файлов Android, так как без него система не может правильно интерпретировать атрибуты Android.

8. Как определить в манифесте иконку приложения?

В манифесте иконка приложения указывается с помощью атрибута android:icon в элементе <application>. Пример:

xml

<application

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name">

</application>

Иконка указывается в виде ссылки на ресурс (например, @drawable/ic\_launcher), который находится в папке res/drawable.

9. Для чего предназначен атрибут android:label элемента <application>в манифесте?

Элемента <activity>?

Атрибут android:label в элементе <application> задает строку, которая будет использоваться как имя приложения, отображаемое на устройстве (например, на экране главного меню).

В элементе <activity> атрибут android:label задает название для этой конкретной активности, которое может быть отображено в заголовке окна или панели инструментов для этой активности. Пример:

xml

<application

android:label="@string/app\_name">

</application>

<activity

android:label="@string/activity\_name">

</activity>

10. Перечислите достоинства и недостатки макета Constraint Layout

пользовательского интерфейса в Adnroid.

Гибкость: Позволяет создавать сложные макеты с помощью ограничений, без необходимости использовать вложенные элементы.

Оптимизация: Снижает количество вложенных элементов, что улучшает производительность интерфейса.

Мощность: Поддерживает множественные виды ограничений для позиционирования элементов, такие как привязка к краям и центральному расположению.

Недостатки:

Сложность: Изначально сложно разобраться в системе ограничений для новичков.

Нагрузки: В некоторых случаях может быть не так оптимально по производительности, как LinearLayout или RelativeLayout, особенно при сложных иерархиях.

Поддержка старых версий: Для использования ConstraintLayout требуется Android Studio и минимальная версия SDK, начиная с API 14.

11. Какие основные шаги необходимо выполнить для подключения репозитория на

github?

Шаг 1: Создать новый репозиторий на GitHub.

Шаг 2: В Android Studio открыть проект и перейти в меню VCS → Enable Version Control Integration, выбрать Git.

Шаг 3: Ввести команду для инициализации локального репозитория (если еще не инициализирован):

git init

Шаг 4: Добавить удаленный репозиторий (URL репозитория на GitHub):

git remote add origin <URL репозитория>

Шаг 5: Сделать первый коммит и отправить его в репозиторий:

git add .

git commit -m "Initial commit"

git push -u origin master

12. Как создать коммит в Android Studio и просмотреть историю коммитов?

Шаг 1: Внесите изменения в файлы проекта.

Шаг 2: Перейдите в VCS → Commit или используйте комбинацию клавиш Ctrl + K (Windows) / Cmd + K (Mac).

Шаг 3: В появившемся окне выберите файлы для коммита, добавьте сообщение и нажмите Commit или Commit and Push.

История коммитов: Для просмотра истории откройте VCS → Git → Show History, чтобы увидеть список всех коммитов.

13. Как создать ветку в репозитории в Android Studio?

Шаг 1: Перейдите в VCS → Git → Branches.

Шаг 2: Выберите New Branch.

Шаг 3: Введите имя ветки и нажмите Create.

Шаг 4: Для переключения между ветками, выберите ветку в том же меню Branches.

14. Как получить изменения из внешнего репозитория и опубликовать их во внешний

репозиторий?

1. Получить изменения из внешнего репозитория (Pull)

Шаг 1: Откройте Android Studio и перейдите в меню VCS (Version Control System) в верхней панели.

Шаг 2: Выберите Git → Pull. Это откроет окно, в котором можно настроить источник для получения изменений.

Шаг 3: Убедитесь, что выбран правильный удаленный репозиторий (обычно это origin) и нужная ветка.

Шаг 4: Нажмите Pull, чтобы забрать последние изменения из удаленного репозитория в ваш локальный.

2. Опубликовать изменения во внешний репозиторий (Push)

Шаг 1: После того как вы внесли изменения в проект, откройте меню VCS → Commit или используйте горячие клавиши Ctrl + K (Windows) / Cmd + K (Mac).

Шаг 2: В окне коммита выберите файлы, которые вы хотите добавить в коммит, и добавьте сообщение.

Шаг 3: Нажмите Commit для создания коммита в вашем локальном репозитории. Если вы хотите сразу отправить изменения в удаленный репозиторий, используйте Commit and Push.

Шаг 4: Если вы выбрали Commit, затем в меню VCS → Git → Push отправьте изменения в внешний репозиторий. В появившемся окне выберите ветку, куда хотите отправить изменения, и нажмите Push.

Примечания:

Pull: Этот процесс позволяет синхронизировать ваш локальный репозиторий с внешним, получая все последние изменения, сделанные другими пользователями.

Push: Этот процесс отправляет ваши локальные изменения в удаленный репозиторий, чтобы другие могли их получить и использовать.

Таким образом, Pull забирает изменения, а Push публикует их.